Making the Top 25 List

25 CWEs + 4 phases + 1 experts/35 organizations = 6 weeks [illegible: overlapping text]

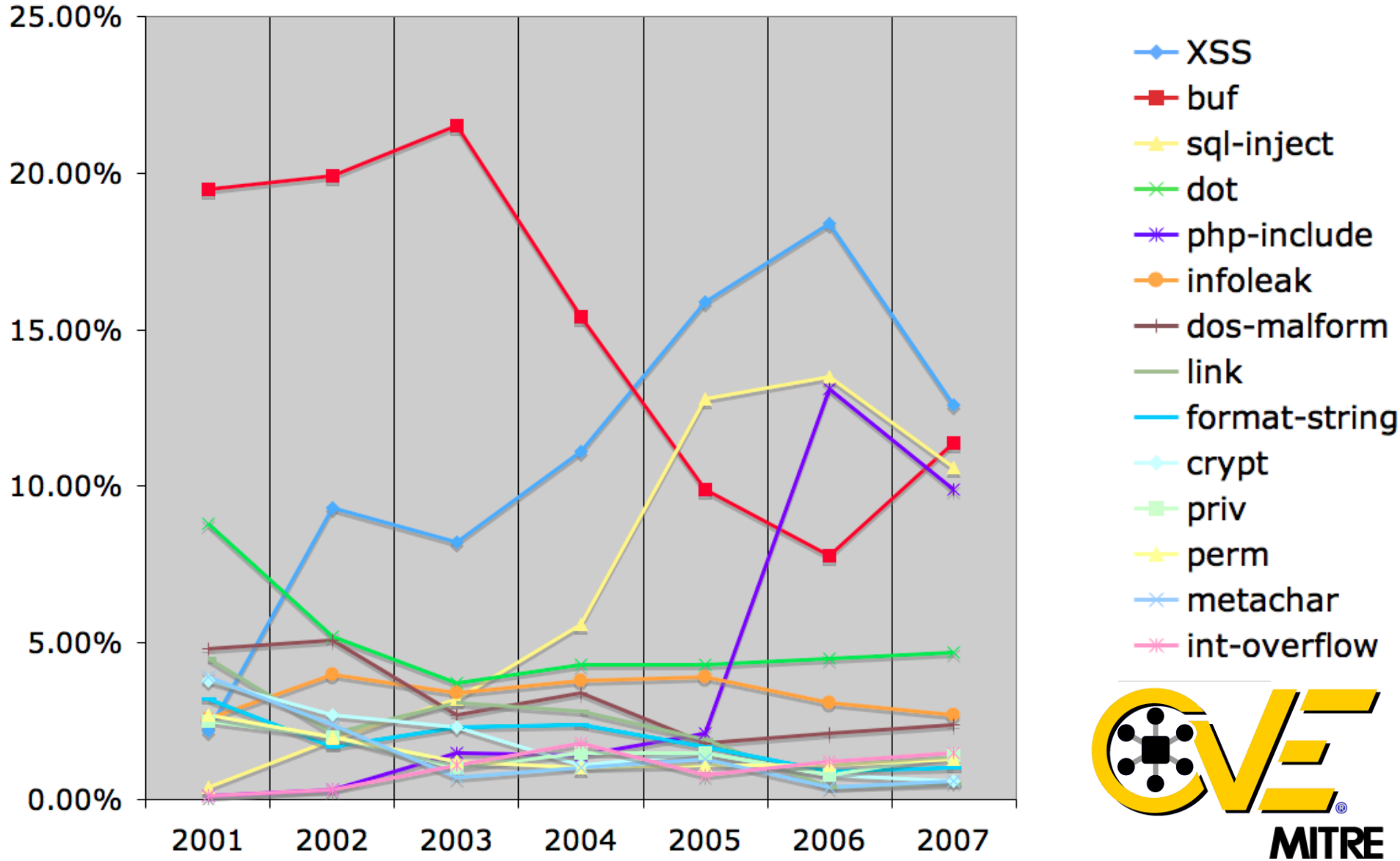
Robert A. Martin

Goal of the Common Weakness Enumeration Initiative

- To improve the quality of software with respect to known security issues within architecture, design, code or implementation. By:
  - **enabling more effective discussion and description of these weaknesses**
  - **defining a unified measurable set of these weaknesses**
  - **supporting the selection and use of software security tools and services to find these weaknesses**

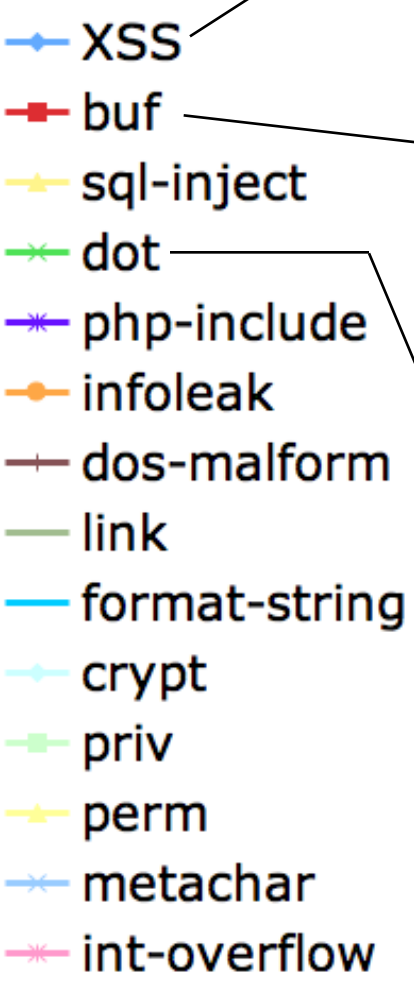
Making Security Measurable™

Vulnerability Type Trends: A Look at the CVE List (2001 - 2007)



CVE®
MITRE

Removing and Preventing the Vulnerabilities Requires More Specific Definitions…CWEs

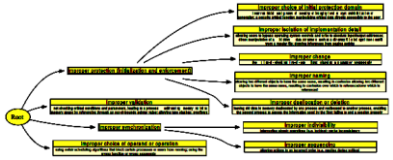


Failure to Sanitize Directives in a Web Page (aka 'Cross-site scripting' (XSS)) (79)
- Failure to Sanitize Script-Related HTML Tags in a Web Page (Basic XSS) (80)
- Failure to Sanitize Directives in an Error Message Web Page (81)
- Failure to Sanitize Script in Attributes of IMG Tags in a Web Page (82)
- Failure to Sanitize Script in Attributes in a Web Page (83)
- Failure to Resolve Encoded URI Schemes in a Web Page (84)
- Doubled Character XSS Manipulations (85)
- Invalid Characters in Identifiers (86)
- Alternate XSS syntax (87)

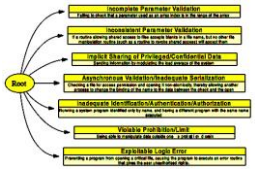
Failure to Constrain Operations within the Bounds of an Allocated Memory Buffer (119)
- Unbounded Transfer (‘Classic Buffer Overflow’) (120)
- Write-what-where Condition (123)
- Boundary Beginning Violation (’Buffer Underwrite') (124)
- Out-of-bounds Read (125)
- Wrap-around Error (128)
- Unchecked Array Indexing (129)
- Incorrect Calculation of Buffer Size (131)
- Miscalculated Null Termination (132)
- Return of Pointer Value Outside of Expected Range (466)

Path Traversal (22)
- Relative Path Traversal (23)
  - Path Traversal: '\..\filename' (29)
  - Path Traversal: '\dir\..\filename' (30)
  - Path Traversal: 'dir\..\filename' (31)
  - Path Traversal: '...' (Triple Dot) (32)
  - Path Traversal: '....' (Multiple Dot) (33)
  - Path Traversal: '..../' (34)
  - Path Traversal: '.../.../' (35)
- Absolute Path Traversal (36)
  - Path Traversal: '/absolute/pathname/here’ (37)
  - Path Traversal: '\absolute\pathname\here’ (38)
  - Path Traversal: 'C:dirname’ (39)
  - Path Traversal: '\\UNC\share\name\' (Windows UNC Share) (40)

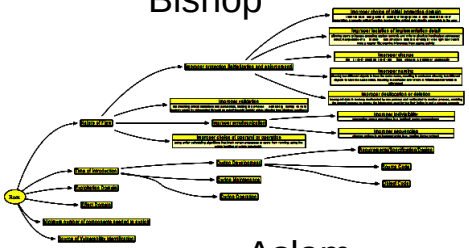
Protection Analysis



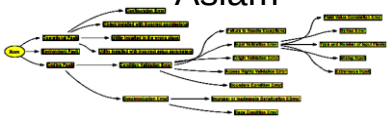
RISOS



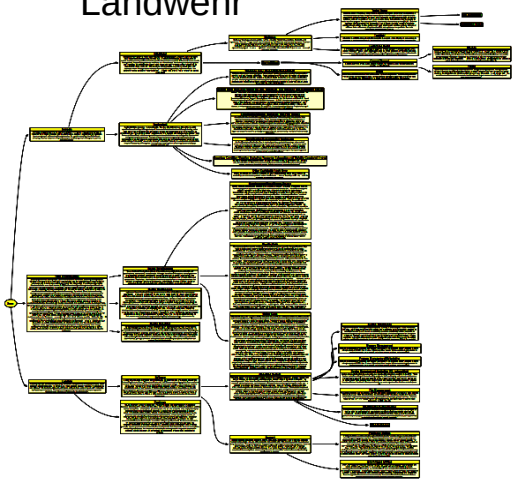
Bishop



Aslam

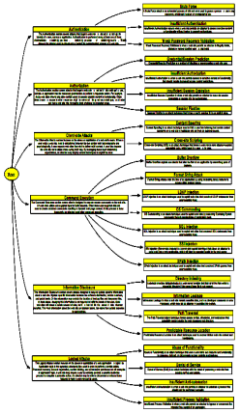


Landwehr



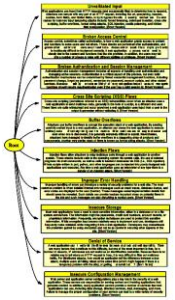
Weber

WASC



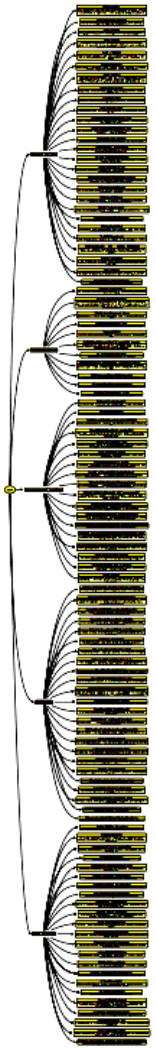
Tool A

OWASP

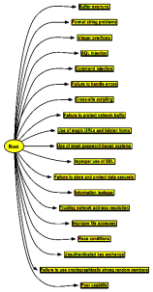


Tool B

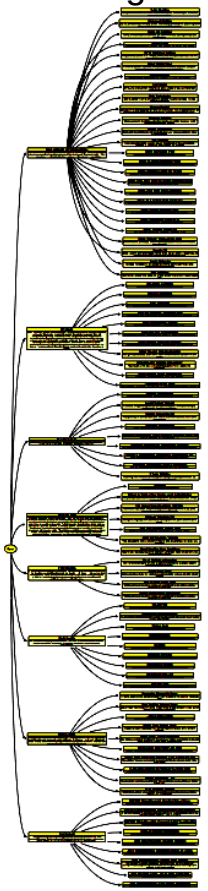
CLASP



Microsoft

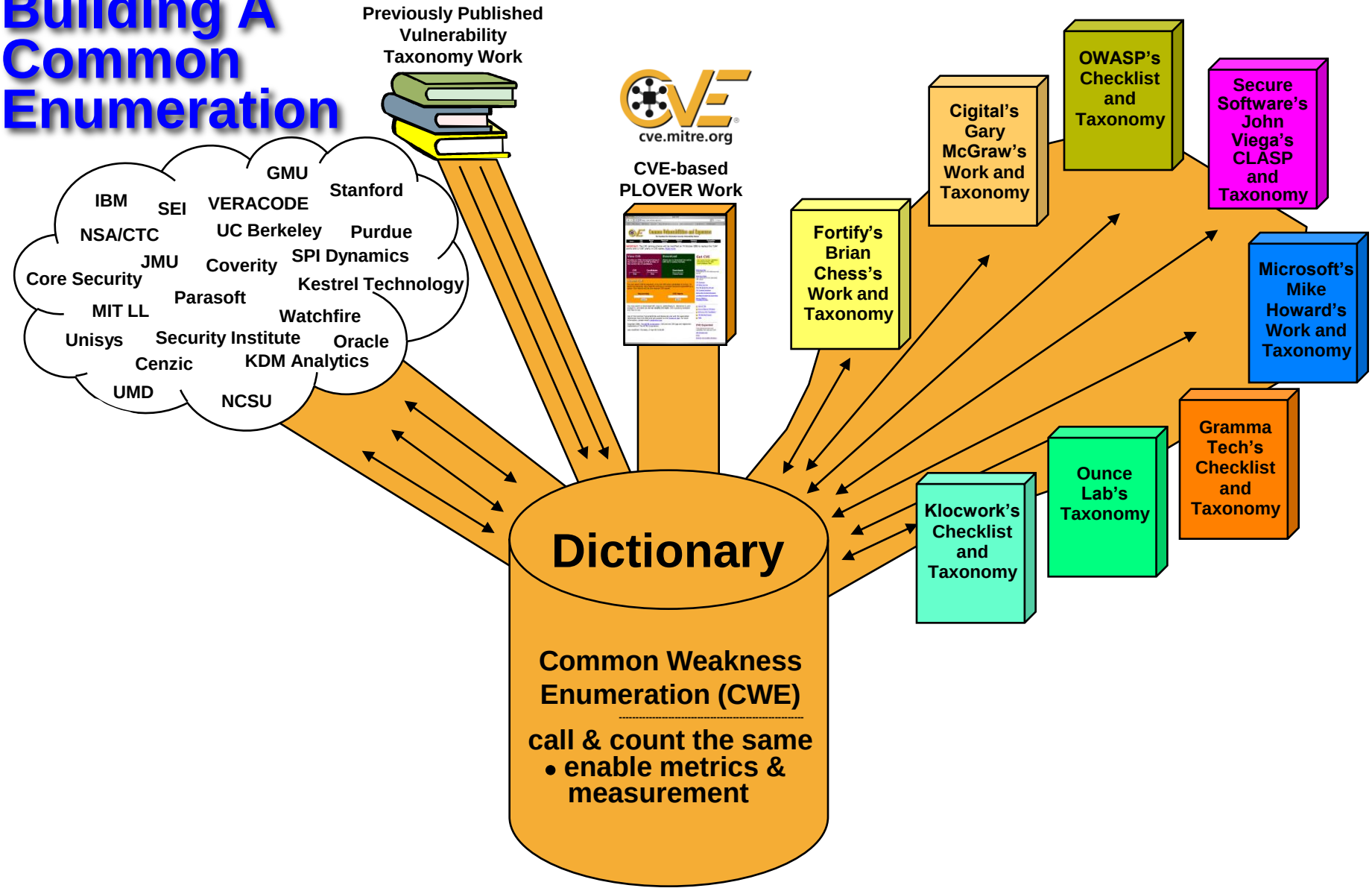


7 Kingdoms



PLOVER

Building A Common Enumeration



Making Security Measurable™

Using A Unilateral NDA with MITRE to Bring in Info

Purpose:

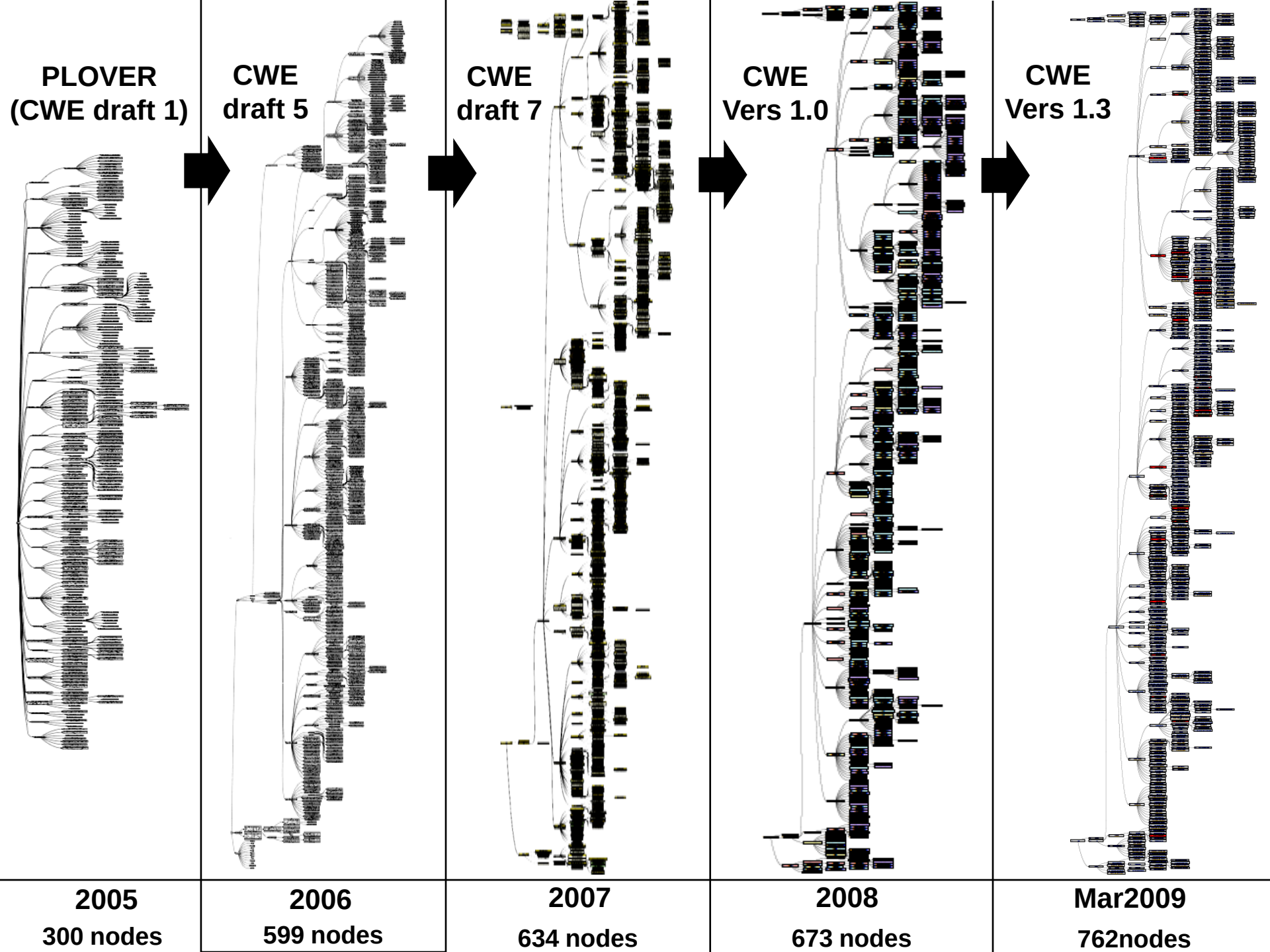
- Sharing the proprietary/company confidential information contained in the underlying Knowledge Repository of the Knowledge Owner's Capability for the sole purpose of establishing a public Common Weakness Enumeration (CWE) dictionary that can be used by vendors, customers, and researchers to describe software, design, and architecture related weaknesses that have security ramifications.
- The individual contributions from numerous organizations, based on their proprietary/company-confidential information, will be combined into a consolidated collection of weakness descriptions and definitions with the resultant collection being shared publicly.
- The consolidated collection of knowledge about weaknesses in software, design, and architecture will make no reference to the source of the information used to describe, define, and explain the individual weaknesses.

Current Community Contributing to the Common Weakness Enumeration

- AppSIC
- Apple
- Cenzic
- Core Security
- Coverity
- Fortify
- Gramma Tech
- Interoperability Clearing House
- KDM Analytics
- Klocwork
- Ounce Labs
- Palamida
- Parasoft
- proServices Corporation
- SecurityInnovation
- SofCheck
- SPI Dynamics
- SureLogic, Inc.
- VERACODE
- Watchfire

Making Security Measurable™

To join send e-mail to cwe@mitre.org


PLOVER
(CWE draft 1)
CWE
draft 5
CWE
draft 7
CWE
Vers 1.0
CWE
Vers 1.3
2005
300 nodes
2006
599 nodes
2007
634 nodes
2008
673 nodes
Mar2009
762nodes

CWE Compatibility & Effectiveness Program

(launched Feb 2007)

armorize technologies
Secure Your Web Code
SECURITY DATABASE
KEEP A VIGILANT EYE ON YOUR DEFENSES
VERACODE
FORTIFY SOFTWARE
SPI DYNAMICS
CENZIC
GRAMMATECH
Klocwork
SANS
SkillBridge
IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
SofCheck
IBM
watchfire
checkmarx
cigital
OUNCE LABS
CERIAS

Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

Products are listed alphabetically by organization name:

TOTALS
Organizations Participating: **18**
Products & Services: **32**

cwe.mitre.org/compatible/

Recent Posts

MS08-078 and the SDL
Announcing CAT.NET CTP and AntiXSS v3 beta
SDL videos
BlueHat SDL Sessions Wrap-up
Secure Coding Secrets?

Tags

Common Criteria **Crawl Walk Run**
Privacy **SDL** SDL Pro Network
Security Assurance Security Blackhat
SDL **threat modeling**

News

Blogroll

BlueHat Security Briefings
The Microsoft Security Response Center
Michael Howard's Web Log
The Data Privacy Imperative
Security Vulnerability Research & Defense
Visual Studio Code Analysis Blog
MSRC Ecosystem Strategy Team

Books / Papers / Guidance

The Security Development Lifecycle (Howard and Lipner)
Privacy Guidelines for Developing Software Products and Services
Microsoft Security Development Lifecycle (SDL) — Portal
Microsoft Security Development Lifecycle (SDL) — Process Guidance (Web)
Microsoft Security Development Lifecycle (SDL) — Process Guidance (.doc)

MS08-078 and the SDL ★★★★★

Hi, Michael here.

Every bug is an opportunity to learn, and the security update that fixed the data binding bug that affected Internet Explorer users is no exception.

The Common Vulnerabilities and Exposures (CVE) entry for this bug is CVE-2008-4844.

Before I get started, I want to explain the goals of the SDL and the security work here at Microsoft. The SDL is designed as a multi-layered process to help systemically reduce security vulnerabilities; if one component of the SDL process fails to prevent or catch a bug, then some other component should prevent or catch the bug. The SDL also mandates the use of security defenses whose impact will be reflected in the "mitigations" section of a security bulletin, because we know that no software development process will catch all security bugs. As we have said many times, the goal of the SDL is to "Reduce vulnerabilities, and reduce the severity of what's missed."

In this post, I want to focus on the SDL-required code analysis, code review, fuzzing and compiler and operating system defenses and how they fared.

Background

The bug was an invalid pointer dereference in MSHTML.DLL when the code handles data binding. It's important to point out that there is no heap corruption and there is no heap-based buffer overrun!

When data binding is used, IE creates an object which contains an array of data binding objects. In the code in question, when a data binding object is released, the array length is not correctly updated leading to a function call into freed memory.

The vulnerable code looks a little like this (by the way, the real array name is _aryPXfer, but I figured ArrayOfObjectsFromIE is a little more descriptive for people not in the Internet Explorer team.)

```
int MaxIdx = ArrayOfObjectsFromIE.Size()-1;
for (int i=0; i <= MaxIdx; i++) {
    if (!ArrayOfObjectsFromIE[i])
        continue;
    ArrayOfObjectsFromIE[i]->TransferFromSource();
    ...
}
```

Here's how the vulnerability manifests itself: if there are two data transfers with the same identifier (so MaxIdx is 2), and the first transfer updates the length of the ArrayOfObjectsFromIE array when its work was done and releases its data binding object, the loop count would still be whatever MaxIdx was at the start of the loop, 2.

This is a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The Common Weakness Enumeration (CWE) classification for this vulnerability is CWE-367.

The fix was to check the maximum iteration count on each loop iteration rather than once before the loop starts; this is the correct fix for a TOCTOU bug - move the check as close as possible to the action because, in

a time-of-check-time-of-use (TOCTOU) bug that led to code calling into a freed memory block. The
on Weakness Enumeration (CWE) classification for this vulnerability is CWE-367.

TOCTOU issues. We will update our training to address this.

Our static analysis tools don't find this because the tools would need to understand the re-entrant nature of the code.

Fuzz Testing

September 2008 (5)
August 2008 (2)
July 2008 (8)
June 2008 (4)

2009 SANS/CWE Top 25 Programming Errors
(released 12 Jan 2009) cwe.mitre.org/top25/

Making Security Measurable™

CWE/SANS Top 25 Programming Errors

- Sponsored by:
  - **National Cyber Security Division (DHS)**
  - **Information Assurance Division (NSA)**
- List was selected by a group of security experts from 35 organizations including:
  - **Academia: Purdue, Univ. of Cal., N. Kentucky Univ.**
  - **Government: CERT, NSA, DHS**
  - **Software Vendors: Microsoft, Oracle, Red Hat, Apple**
  - **Security Vendors: Veracode, Fortify, Cigital, Symantec**

Making Security Measurable™

Credited Contributors

The following people or organizations are being publicly acknowledged because they provided us with substantive comments on the drafts. This public document is markedly improved thanks to their expert feedback.

Additionally, without the advice and collaboration from Alan Paller and Mason Brown from the SANS Institute, this effort would not be what it has become. Finally, CWE Team members Conor Harris and Janis Kenderdine deserve our endless thanks for their tireless and timely help in updating the CWE items and getting this material into a usable form on the web site.

Robert A. Martin & Steve Christey

Making Security Measurable™

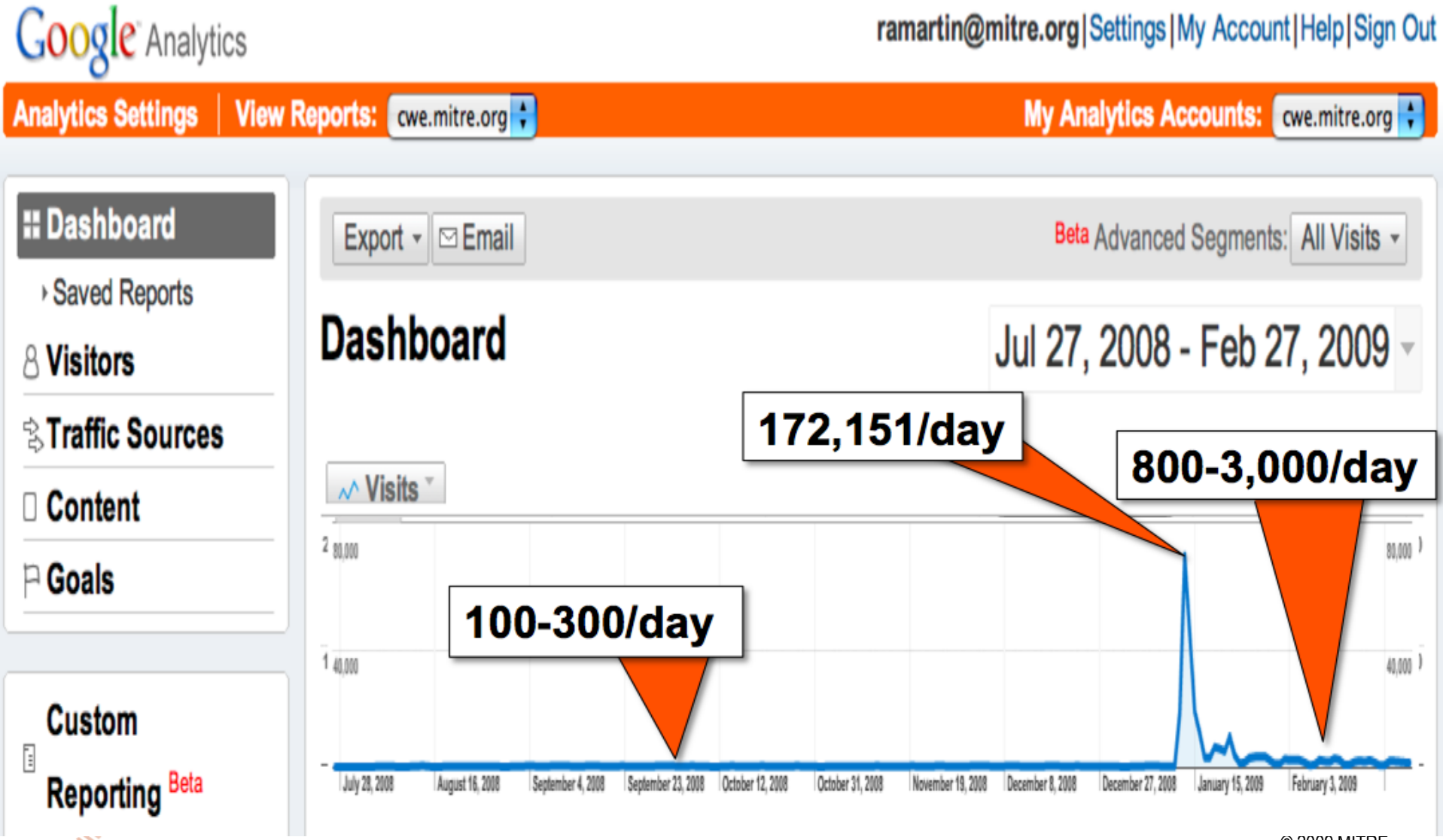
Robert C. Seacord	CERT
Pascal Meunier	CERIAS, Purdue University
Matt Bishop	University of California, Davis
Kenneth van Wyk	KRvW Associates
Masato Terada	Information-Technology Promotion Agency (IPA) (Japan)
Sean Barnum	Cigital, Inc.
Mahesh Saptarshi	Symantec Corporation
Cassio Goldschmidt	Symantec Corporation
Adam Hahn	MITRE
Jeff Williams	Aspect Security and OWASP
Carsten Eiram	Secunia
Josh Drake	iDefense Labs at VeriSign, Inc.
Chuck Willis	MANDIANT
Michael Howard	Microsoft
Bruce Lowenthal	Oracle Corporation
Mark J. Cox	Red Hat Inc.
Jacob West	Fortify Software
Djenana Campara	Hatha Systems
James Walden	Northern Kentucky University
Frank Kim	ThinkSec
Chris Eng	Veracode, Inc.
Chris Wysopal	Veracode, Inc.
Ryan Barnett	Breach Security
Antonio Fontes	New Access SA (Switzerland)
Mark Fioravanti II	Missing Link Security Inc.
Ketan Vyas	Tata Consultancy Services (TCS)
Lindsey Cheng	Secured Sciences Group, LLC
Ian Peters	Secured Sciences Group, LLC
Tom Burgess	Secured Sciences Group, LLC
Hardik Parekh	RSA - Security Division of EMC Corporation
Matthew Coles	RSA - Security Division of EMC Corporation
Mouse	
Ivan Ristic	
Apple Product Security	
Software Assurance Forum for Excellence in Code (SAFECode)	
Core Security Technologies Inc.	
Depository Trust & Clearing Corporation (DTCC)	
The working group at the first OWASP ESAPI Summit	
National Security Agency (NSA) Information Assurance Division	
Department of Homeland Security (DHS) National Cyber Security Division	

Main Goals

- Raise awareness for developers
- Help universities to teach secure coding
- Empower customers who want to ask for more secure software
- Provide a starting point for in-house software shops to measure their own progress

Making Security Measurable™

People are Starved for Simplicity



Who Did We Reach and Where?

- News: USA Today, Forbes, BBC
- Trade Magazines
- Blogs, tweets, bookmarks
- Podcasts
- Developers
- Friends, Romans, Countrymen

Making Security Measurable™

Some Reactions (Paraphrased)

- Blog title: “NSA flames N00bs”
- “I never heard of any of these. Thanks!”
- “I have a feeling I’ll be busy this weekend.”
- “You forgot #1: managers force us to meet deadlines.”
- “My boss asked what I thought about this.”
- “It’s convenient to have these all in one place”
- “This complicates my job as a consultant”
- “This one is easy to fix.” “No it’s not!” “Oh, yeah.”
- “These are all just (web problems|injection|bugs)”
- [in vendor forum]
  - **Customer: “How have you protected against these?”**
  - **Vendor: <silence>**

Making Security Measurable™

Prevalence based on 2008 CVE data

Category	Count	%
SQL Injection	941	19.4%
XSS	681	14.0%
Buffer Overflow	455	9.4%
Directory Traversal	298	6.1%
PHP Include	135	2.8%
Symbolic Link	133	2.7%
Authorization Bypass	113	2.3%
Doubly-formed Input	97	2.0%
Information Leak	81	1.7%
Integer Overflow	78	1.6%
CSRF	57	1.2%
Bad Permissions	40	0.8%
Unnecessary Privileges	36	0.7%

Hard coded Password	36	0.7%
Upload of code	34	0.7%
Weak Crypto	30	0.6%
Format String	26	0.5%
Insufficient Randomness	24	0.5%
Metacharacter Injection	23	0.5%
Session Fixation	20	0.4%
Memory Leak	18	0.4%
Sensitive data read	16	0.3%
Race Condition	15	0.3%
DoS Flood	10	0.2%
CRLF Injection	8	0.2%
Eval Injection	8	0.2%
Numeric Error	7	0.1%

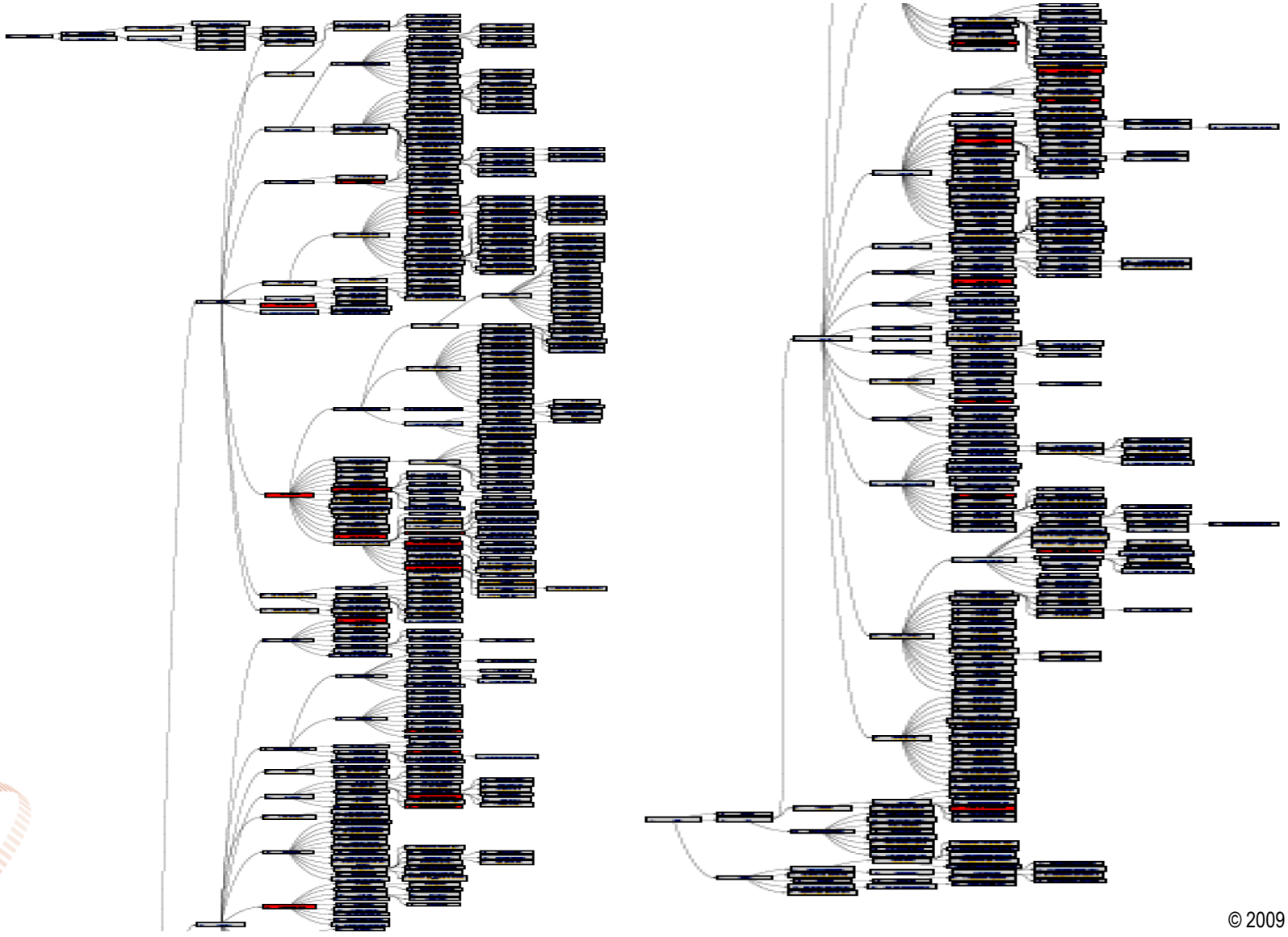
Plus Info from Various Consultants Regarding "Internally Developed Code"

INCOMPLETE DATA

Making Security Measurable™

4855 total flaws tracked by CVE in 2008

Fear the Rest:
The Top 25 compared to all CWE



Making Security Measurable™

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output
- CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')
- CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')
- CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-362: Race Condition
- CWE-209: Error Message Information Leak

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

- CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer
- CWE-642: External Control of Critical State Data
- CWE-73: External Control of File Name or Path
- CWE-426: Untrusted Search Path
- CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
- CWE-494: Download of Code Without Integrity Check
- CWE-404: Improper Resource Shutdown or Release
- CWE-665: Improper Initialization
- CWE-682: Incorrect Calculation

Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

- CWE-285: Improper Access Control (Authorization)
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-259: Hard-Coded Password
- CWE-732: Insecure Permission Assignment for Critical Resource
- CWE-330: Use of Insufficiently Random Values
- CWE-250: Execution with Unnecessary Privileges
- CWE-602: Client-Side Enforcement of Server-Side Security

Background Details to Check Out

cwe.mitre.org/top25

- Contributors
- Process description
- Changelog for each revision
- On the Cusp – weaknesses that almost made it
- Appendices
  - **Selection Criteria and Supporting Fields**
  - **Threat Model for the Skilled, Determined Attacker**

Making Security Measurable™

2009 CWE/SANS Top 25 - Final Draft Changes and Discussion

Changes in Final Week

Date: January 11, 2009

- Improved readability and understandability of discussion text
- Added remainder of suggested mitigations to CWE entries
- Made significant updates to CWE entries on the Top 25, focusing on demonstrative examples, mitigations, consequences, references, and extended descriptions.
- Finished additions to the "On the Cusp" list of CWEs that did not make it to the Top 25
- Created a CWE Top 25 view (CWE-750) and generated supporting PDF graphs for visualization
- Collected final supporting quotes
- Wrote process document
- Finalized contributor list
- Reorganized main document

BACK TO TOP

Summary of Received Comments for Draft 3

Date: January 8, 2009

- We again received many comments from about a dozen people, so we cannot individually respond to them all. Each draft has had approximately 5 to 8 new reviewers.
- Many of the comments were related to specific mitigations for individual entries. The CWE entries are being updated to reflect these changes.
- Some people provided substantive commentary on the threat model, which was a new addition. Much of the feedback centered around apparent contradictions and other issues with the description. As a result, this was cleaned up a bit in this final draft.
- Many contributors made final requests for entries that they thought were important for inclusion. In some cases, there were conflicting recommendations from different people, especially with respect to prevalence. The same entry would be seen extensively by one person but much less frequently by another person. This made the final decisions more difficult. A separate "On the Cusp" document will be published that will cover the issues that did not make it to the final list.

Top 25 and OWASP Top 10

C	Causal (Chain)	***	Exact Match
~	Indirect Relationship	^	Abstraction Relationship

also see: http://securityninja.co.uk/blog/?p=132

	20	116	89	79	78	319	352	362	209	119	642	73	426	94	494	404	665	682	285	327	259	732	330	250	602
A1: XSS	C	C		***							C														C
A2: Injection	C	C	^	^	^						C		~	***											C
A3: Malicious File Execution	C										C	***		~			C			C					C
A4: Direct Object Reference	~										***	~													C
A5: CSRF				C			***																C		C
A6: Information Leaks / Error Handling									^																C
A7: Authentication / Session Management											C										^		C		C
A8: Cryptographic Storage																							C		
A9: Communications						***																			
A10: URL Restriction																			^			C	C		~

RE

Frequently Asked Questions (FAQ)

How is this different from the OWASP Top Ten?

The short answer is that the OWASP Top Ten covers more general concepts and is focused on web applications. The CWE Top 25 covers a broader range of issues than what arise from the web-centric view of the OWASP Top Ten, such as buffer overflows. Also, one goal of the CWE Top 25 is to be at a level that is directly actionable to programmers, so it contains more detailed issues than the categories being used in the Top Ten. There is some overlap, however, since web applications are so prevalent, and some issues in the Top Ten have general applications to all classes of software.

How are the weaknesses prioritized on the list?

With the exception of Input Validation being listed as number 1 (partially for educational purposes), there is no concrete prioritization. Prioritization differs widely depending on the audience (e.g. web application developers versus OS developers) and the risk tolerance (whether code execution, data theft, or denial of service are more important). It was also believed that the use of categories would help the organization of the document, and prioritization would impose a different ordering.

Why are you including overlapping concepts like input validation and XSS, or incorrect calculation and buffer overflows? Why do you have mixed levels of abstraction?

While it would have been ideal to have a fixed level of abstraction and no overlap between weaknesses, there are several reasons why this was not achieved.

Contributors sometimes suggested different CWE identifiers that were closely related. In some cases, this difference was addressed by using a more abstract CWE identifier that covered the relevant cases.

In other situations, there was strong advocacy for including lower-level issues such as SQL injection and cross-site scripting, so these were added. The general trend, however, was to use more abstract weakness types.

While it might be desired to minimize overlap in the Top 25, many vulnerabilities actually deal with the interaction of 2 or more weaknesses. For example, external control of user state data (CWE-642) could be an important weakness that enables cross-site scripting (CWE-79) and SQL injection (CWE-89). To eliminate overlap in the Top 25 would lose some of this important subtlety.

Finally, it was a conscious decision that if there was enough prevalence and severity, design-related weaknesses would be included. These are often thought of as being more abstract than weaknesses that arise during implementation.

The Top 25 list tries to strike a delicate balance between usability and relevance, and we believe that it does so, even with this apparent imperfection.

Why don't you use hard statistics to back up your claims?

The appropriate statistics simply aren't publicly available. The publicly available statistics are either too high-level or not comprehensive enough. And none of them are comprehensive across all software types and environments.

Fear #26… both of 'em

- Resource Exhaustion
  - **Not prevalent enough**
  - **Not severe enough**
    - Based on T25's threat model
- Unchecked Return Value
  - **Not prevalent enough**
  - **Rarely severe enough**
- What's your #26?

… as far as we know

Making Security Measurable™

On the Cusp: Other Weaknesses to Consider

Table of Contents

Introduction

The CWE/SANS Top 25 is really just a starting point for developers. Many weaknesses were considered for inclusion on the Top 25, but some did not make it to the final list. Some were not considered to be severe enough; others were not considered to be prevalent enough. Sometimes, the Top 25 reviewers themselves had mixed opinions on whether a weakness should be added to the list or not.

With respect to severity, some Top 25 users may have a significantly different threat model. For example, software uptime may be critical to consumers who operate in critical infrastructure or e-commerce environments. However, in the threat model being used by the Top 25, availability is regarded as slightly less important than integrity and confidentiality.

With respect to prevalence, some Top 25 items may not be applicable to the class of software being developed. For example, cross-site scripting is specific to the Web, although analogs exist in other technologies. In other cases, developers may have already eliminated much of the Top 25 in past efforts, so they want to look for other weaknesses that may still be present in their software.

Some on-the-cusp items were omitted because they are already indirectly covered on the Top 25, usually by a more general entry. However, these would be important to consider as individual items.

For these reasons, users of the Top 25 should seriously consider including these weaknesses in their analyses.

BACK TO TOP

Weaknesses that did not have sufficient prevalence or severity

NSA, DHS, Industry Gang Up on Dangerous Software Errors

Posted by: Stephen Wildstrom on January 12

Computer security experts have warned for years that the endless cycle of software flaws and exploits will only be broken when we create incentives for software authors and publishers to get it right. On Jan. 12, the industry took a potentially important step toward that goal when a broad coalition of companies, government agencies, academics, and advocacy groups launched a program to assure than software is free of 25 common errors that lead to the bulk of security problems.

The key to making the program effective is that it goes well beyond recommending best practices. Software buyers, particularly governments and large corporations are being urged to demand that vendors certify that code they sell is free of these 25 errors, and there's nothing like potential legal liability to get a company's attention. In addition, colleges are pledging to train students in writing software and employers can use the guidelines to assess the skills of

Recent Posts

SDL Threat Modeling Tool 3.1.4 ships!
Early Days of the SDL, Part Four
Early Days of the SDL, Part Three
Early Days of the SDL, Part Two
Early Days of the SDL, Part One

Tags

Common Criteria Crawl Walk Run Privacy SDL SDL Pro Network Security Assurance Security Blackhat SDL threat modeling

News

About Us

Adam Shostack
Bryan Sullivan
David Ladd
Jeremy Dallman
Michael Howard
Steve Lipner

Blogroll

BlueHat Security Briefings

SDL and the CWE/SANS Top 25

Bryan here. The security community has been buzzing since SANS and MITRE's joint announcement earlier this month of their list of the Top 25 Most Dangerous Programming Errors. Now, I don't want to get into a debate in this blog about whether this new list will become the new de facto standard for analyzing security vulnerabilities (or indeed, whether it already has become the new standard). Instead, I'd like to present an overview of how the Microsoft SDL maps to the CWE/SANS list, just
May.

Michael and I have writte
coverage of the Top 25 ar
believe that the results te
25 were developed indep
root them out of the softv
analysis white paper and
guidance around every m
made many of the same S
for you to download and u

Below is a summary of ho
see the SDL covers every
them (race conditions and
by multiple SDL requirem
tools to prevent or detect

CWE	Title
20	Improper Input Va
116	Improper Encodin Escaping of Outpu

CWE	Title	Education?	Manual Process?	Tools?	Threat Model?
20	Improper Input Validation	Y	Y	Y	Y
116	Improper Encoding or Escaping of Output	Y	Y	Y	
89	Failure to Preserve SQL Query Structure (aka SQL Injection)	Y	Y	Y	
79	Failure to Preserve Web Page Structure (aka Cross-Site Scripting)	Y	Y	Y	
78	Failure to Preserve OS Command Structure (aka OS Command Injection)	Y		Y	
319	Cleartext Transmission of Sensitive Information	Y			Y
352	Cross-site Request Forgery (aka CSRF)	Y		Y	
362	Race Condition	Y			
209	Error Message Information Leak	Y	Y	Y	
119	Failure to Constrain Memory Operations within the Bounds of a Memory Buffer	Y	Y	Y	
642	External Control of Critical State Data	Y			Y
73	External Control of File Name or Path	Y	Y	Y	
426	Untrusted Search Path	Y		Y	
94	Failure to Control Generation of Code (aka 'Code Injection')	Y	Y		
494	Download of Code Without Integrity Check				Y
404	Improper Resource Shutdown or Release	Y		Y	
665	Improper Initialization	Y		Y	
682	Incorrect Calculation	Y		Y	
285	Improper Access Control (Authorization)	Y	Y		Y
327	Use of a Broken or Risky Cryptographic Algorithm	Y	Y	Y	
259	Hard-Coded Password	Y	Y	Y	Y
732	Insecure Permission Assignment for Critical Resource	Y	Y		
330	Use of Insufficiently Random Values	Y	Y	Y	
250	Execution with Unnecessary Privileges	Y	Y		Y
602	Client-Side Enforcement of Server-Side Security	Y			Y

The Top 25 is not…

- A silver bullet
- A guarantee of software health
- A perfect match for your unique needs
- As simple as it seems
- The only thing to include in contract language
- Completely found by tools

Making Security Measurable™

The Top 25 is…

- A mechanism for awareness
- A trigger of questions
- A place for mitigations
- A conversation starter
- A first step on the long road to software assurance

Making Security Measurable™

Contact Us

top25@sans.org

cwe@mitre.org

cwe.mitre.org/top25

Public discussion list coming soon

http://www.owasp.org/index.php/Podcast_11

Making Security Measurable™